Оглавление

[Введение 2](#_Toc197801565)

[1. Разработка методов дискретного логарифмирования 6](#_Toc197801566)

[2. Вспомогательные математические функции 8](#_Toc197801567)

[3. Алгоритм Шенкса 12](#_Toc197801568)

[4. Алгоритм Полига-Хеллмана 20](#_Toc197801569)

[5. Алгоритм ро-метод Полларда 28](#_Toc197801570)

[6. Алгоритм Адлемана 40](#_Toc197801571)

[7. Алгоритм COS 48](#_Toc197801572)

[8. Алгоритм решето числового поля 57](#_Toc197801573)

[Заключение 70](#_Toc197801574)

[Список литературы 72](#_Toc197801575)

[Приложения 73](#_Toc197801576)

# Введение

Дискре́тное логарифми́рование (DLOG) — задача обращения функции в некоторой конечной мультипликативной группе .

Наиболее часто задачу дискретного логарифмирования рассматривают в мультипликативной группе кольца вычетов или конечного поля, а также в группе точек эллиптической кривой над конечным полем. Эффективные алгоритмы для решения задачи дискретного логарифмирования в общем случае неизвестны.

Для заданных g и a решение x уравнения называется дискретным логарифмом элемента a по основанию g. В случае, когда G является мультипликативной группой кольца вычетов по модулю m, решение называют также индексом числа a по основанию g. Индекс числа a по основанию g гарантированно существует, если g является первообразным корнем по модулю m.

Пусть в некоторой конечной мультипликативной абелевой группе задано уравнение

Решение задачи дискретного логарифмирования состоит в нахождении некоторого целого неотрицательного числа , удовлетворяющего уравнению . Если оно разрешимо, у него должно быть хотя бы одно натуральное решение, не превышающее порядок группы. Это сразу даёт грубую оценку сложности алгоритма поиска решений сверху — алгоритм полного перебора нашёл бы решение за число шагов не выше порядка данной группы.

Чаще всего рассматривается случай, когда , то есть группа является циклической, порождённой элементом . В этом случае уравнение всегда имеет решение. В случае же произвольной группы вопрос о разрешимости задачи дискретного логарифмирования, то есть вопрос о существовании решений уравнения (1), требует отдельного рассмотрения.

Рассмотрим задачу дискретного логарифмирования в кольце вычетов по модулю простого числа. Пусть задано сравнение

Будем решать задачу методом перебора. Выпишем таблицу всех степеней числа 3. Каждый раз мы вычисляем остаток от деления на 17 (например, 33≡27 — остаток от деления на 17 равен 10).

31 ≡ 3 32 ≡ 9 33 ≡ 10 34 ≡ 13 35 ≡ 5 36 ≡ 15 37 ≡ 11 38 ≡ 16

39 ≡ 14 310 ≡ 8 311 ≡ 7 312 ≡ 4 313 ≡ 12 314 ≡ 2 315 ≡ 6 316 ≡ 1

Теперь легко увидеть, что решением рассматриваемого сравнения является x = 4, поскольку 34≡13.

На практике модуль обычно является достаточно большим числом, и метод перебора является слишком медленным, поэтому возникает потребность в более быстрых алгоритмах.

Существует множество алгоритмов для решения задачи дискретного логарифмирования в поле вычетов. Их принято разделять на экспоненциальные и субэкспоненциальные. Полиномиального алгоритма для решения этой задачи пока не существует.

Алгоритмы с экспоненциальной сложностью:

1. Алгоритм Шенкса (алгоритм больших и малых шагов, baby-step giant-step)

2. Алгоритм Полига - Хеллмана работает, если известно разложение числа на простые множители. Сложность: . Если множители, на которые раскладывается , достаточно маленькие, то алгоритм очень эффективен[2].

3. ρ-Метод Полларда имеет эвристическую оценку сложности .

Субэкспоненциальные алгоритмы:

В L-нотации вычислительная сложность данных алгоритмов оценивается как арифметических операций, где — некоторые константы. Эффективность алгоритма во многом зависит от близости величин и к 1 и 0, соответственно.

1. Алгоритм Адлемана появился в 1979 году[4]. Это был первый субэкспоненциалный алгоритм дискретного логарифмирования. На практике он всё же недостаточно эффективен. В этом алгоритме .

2. Алгоритм COS был предложен в 1986 году математиками Копперсмитом (Don Coppersmith), Одлыжко (Andrew Odlyzko) и Шреппелем (Richard Schroeppel)[5]. В этом алгоритме константа , . В 1991 году с помощью этого метода было проведено логарифмирование по модулю . В 1997 году Вебер провёл дискретное логарифмирование по модулю с помощью некоторой версии данного алгоритма. Экспериментально показано, что при алгоритм COS лучше решета числового поля.

3. Дискретное логарифмирование при помощи решета числового поля было применено к дискретному логарифмированию позднее, чем к факторизации чисел. Первые идеи появились в 1990-х годах. Алгоритм, предложенный Д. Гордоном в 1993 году, имел эвристическую сложность , но оказался достаточно непрактичным. Позднее было предложено множество различных улучшений данного алгоритма. Было показано, что при решето числового поля быстрее, чем COS. Современные рекорды в дискретном логарифмировании получены именно с помощью этого метода.

Наилучшими параметрами в оценке сложности на данный момент является .

Для чисел специального вида результат можно улучшить. В некоторых случаях можно построить алгоритм, для которого константы будут , . За счёт того, что константа достаточно близка к 1, подобные алгоритмы могут обогнать алгоритм с .

Актуальность выпускной работы заключается в том, что задача дискретного логарифмирования является одной из основных задач, на которых базируется криптография с открытым ключом. Классическими криптографическими схемами на её основе являются схема выработки общего ключа Диффи-Хеллмана, схема электронной подписи Эль-Гамаля, криптосистема Мэсси-Омуры для передачи сообщений. Их криптостойкость основывается на предположительно высокой вычислительной сложности обращения показательной функции. Хотя сама показательная функция вычисляется достаточно эффективно, даже самые современные алгоритмы вычисления дискретного логарифма имеют очень высокую сложность, которая сравнима со сложностью наиболее быстрых алгоритмов разложения чисел на множители.

Другая возможность эффективного решения задачи вычисления дискретного логарифма связана с квантовыми вычислениями. Теоретически доказано, что с помощью алгоритма Шора дискретный логарифм можно вычислить за полиномиальное время. В любом случае, если полиномиальный алгоритм вычисления дискретного логарифма будет реализован, это будет означать практическую непригодность криптосистем на его основе для долговременной защиты данных. Рассматривается ряд идей для создания новых алгоритмов с открытым ключом.

Целью выпускной работы является исследование и разработка алгоритмов дискретного логарифмирования с экспоненциальной и субэкспоненциальной сложностью. Также исследование и разработка модифицированных алгоритмов на основе разработанных базовых алгоритмов дискретного логарифмирования, проведение экспериментов и сравнение базовых и модифицированных алгоритмов.

Задачами выпускной работы являются:

1) реализовать вспомогательные математические функции для проверки алгоритмов дискретного логарифмирования,

2) исследовать и реализовать базовые алгоритмы дискретного логарифмирования,

3) исследовать и реализовать модифицированные алгоритмы дискретного логарифмирования,

4) провести эксперименты и сравнительный анализ на реализованных базовых и модифицированных методах дискретного логарифмирования.

# 1. Разработка методов дискретного логарифмирования

В процессе выпускной работы были изучены и реализованы базовые и модифицированные методы дискретного логарифмирования на языке программирования C# на .NET8 в Windows Forms (рисунок 1). Для тестирования данных алгоритмов был реализован генератор параметров Диффи-Хеллмана и возведение числа в степень по модулю [1]. Также для тестирования данных алгоритмов был использован замер времени выполнения алгоритма и количество затраченной памяти на выполнение алгоритма.



Рисунок 1 - Реализованная программа

Были реализованы экспоненциальные алгоритмы дискретного логарифмирования: алгоритм Шенкса [2], алгоритм Полига-Хеллмана [3], ро-метод Полларда [4], а также субэкспоненциальные алгоритмы дискретного логарифмирования: алгоритм Адлемана [5], алгоритм COS [6], решето числового поля [7]. Для разработки всех алгоритмов были реализованы вспомогательные математические функции.

Разработанная программа позволяет вносить в текстовые поля необходимые значения параметров возведения чисел в степень по модулю: g, a, p, A, либо целых чисел N для разложения на простые множители и выводить результат вычисления. Для корректности работы программы была реализована проверка на корректность ввода параметров для вычисления результатов алгоритмов. При помощи встроенного метода TryParse() в платформе для разработки программного обеспечения .NET идёт попытка конвертировать введённые значения в BigInteger. Если конвертация проходит успешно, то идёт проверка отрицательность конвертированных значений. Если проверка введённых значений проходит успешно, то идёт вычисление алгоритма (рисунок 2).



Рисунок 2 - Вычисление модифицированных алгоритмов

# 2. Вспомогательные математические функции

Для реализации алгоритмов дискретного логарифмирования были реализованы вспомогательные математические функции.

Была реализована функция быстрого возведения в степень по модулю. Возведение в степень по модулю — это операция над натуральными числами возведения в степень, выполняемая по модулю. Находит применение в информатике, особенно, в области криптографии с открытым ключом. Возведение в степень по модулю — это вычисление остатка от деления натурального числа (основание), возведенного в степень (показатель степени), на натуральное число (модуль). Обозначается: .

Если , и неотрицательны и , то единственное решение существует, причем . Возведение в степень по модулю может быть выполнено и с отрицательным показателем степени . Для этого необходимо найти число , обратное числу по модулю . Это легко сделать с помощью алгоритма Евклида. Таким образом,

.

Возвести в степень по модулю довольно легко, даже при больших входных значениях. А вычисление дискретного логарифма, то есть нахождение показателя степени при заданных , и , намного сложнее. Такое одностороннее поведение функции делает её кандидатом для использования в криптографических алгоритмах.

Шаги алгоритма быстрого возведения в степень по модулю:

1) Показатель степени переводится в двоичный вид.

2) Создаётся список для чисел, первый элементом которого является .

3) В созданный список циклично добавляются остатки от деления квадратов последних элементов на .

4) Все элементы списка, индексы которых совпадают с позициями 1 в двоичном числе из первого пункта, перемножаются.

5) Вычисляется остаток от деления перемноженных чисел списка на .

Для проверки работы разработанного алгоритма были сгенерированы параметры Диффи-Хеллмана: . В результате работы алгоритма был корректно вычислен результат (рисунок 3).

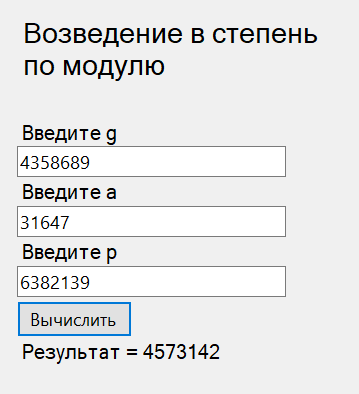


Рисунок 3 - Результат возведения в степень по модулю

Для проверки чисел на простоту был реализован тест Миллера-Рабина [8]. Данный тест является вероятностным полиномиальным тестом простоты. Тест Миллера-Рабина, наряду с тестом Ферма [9] и тестом Соловея-Штрассена [10], позволяет эффективно определить, является ли данное число составным. Однако, с его помощью нельзя строго доказать простоту числа. Тем не менее тест Миллера-Рабина часто используется в криптографии для получения больших случайных простых чисел.

Так как криптостойкость многих алгоритмов шифрования основывается на секретных ключах, для создания которых необходимы простые числа (например, так работает шифр RSA), то при создании таких ключей важно уметь достаточно быстро проверять большие числа на простоту. Вероятностные тесты простоты, такие как тест Миллера-Рабина и тест Соловея-Штрассена, показывают большую эффективность использования и простоту выражения по сравнению с детерминированными тестами. Алгоритм Миллера-Рабина позволяет выполнять проверку за малое время и давать при этом достаточно малую вероятность того, что число на самом деле является составным.

Как и тесты Ферма и Соловея-Штрассена, тест Миллера-Рабина опирается на проверку ряда равенств, которые выполняются для простых чисел. Если хотя бы одно такое равенство не выполняется, это доказывает, что число составное.

Для теста Миллера-Рабина используется следующее утверждение. Пусть – простое число и , где – нечётно. Тогда для любого из выполняется хотя бы одно из условий:

1) .

2) Существует целое число такое, что .

Если это утверждение (условие 1 или 2) выполняется для некоторых чисел и (не обязательно простого), то число называют свидетелем простоты числа по Миллеру, а само число — вероятно простым. (При случайно выбранном вероятность ошибочно принять составное число за простое составляет 25 %, но её можно уменьшить, выполнив проверки для других .)

В случае, когда выполняется контрапозиция доказанного утверждения, то есть если найдётся число такое, что:

и

,

то число не является простым. В этом случае число называют свидетелем того, что число составное.

У нечётных составных чисел существует, согласно теореме Рабина, не более свидетелей простоты, где — функция Эйлера, таким образом вероятность того, что случайно выбранное число окажется свидетелем простоты, меньше 1/4. Идея теста заключается в том, чтобы проверять для случайно выбранных чисел , являются ли они свидетелями простоты числа . Если найдётся свидетель того, что число составное, то число действительно является составным. Если было проверено чисел, и все они оказались свидетелями простоты, то число считается простым. Для такого алгоритма вероятность принять составное число за простое будет меньше .

Для проверки больших чисел принято выбирать числа случайными, так как распределение свидетелей простоты и свидетелей составного числа среди чисел заранее неизвестно.

Алгоритм Миллера-Рабина параметризуется количеством раундов . Рекомендуется брать порядка величины , где — проверяемое число.

Для данного находятся такие целое число и целое нечётное число , что . Выбирается случайное число . Если не является свидетелем простоты числа , то выдаётся ответ « — составное», и алгоритм завершается. Иначе, выбирается новое случайное число и процедура проверки повторяется. После нахождения свидетелей простоты, выдаётся ответ « — вероятно простое», и алгоритм завершается.

# 3. Алгоритм Шенкса

Были реализованы алгоритмы и проведены тесты базового и модифицированного алгоритма «Шаг младенца - шаг великана» - в теории групп детерминированный алгоритм дискретного логарифмирования в мультипликативной группе кольца вычетов по модулю простого числа. Начальный алгоритм был предложен советским математиком Александром Гельфондом в 1962 году и Дэниелом Шенксом в 1972 году. Метод теоретически упрощает решение задачи дискретного логарифмирования, на вычислительной сложности которой построены многие криптосистемы с открытым ключом. Относится к методам встречи посередине. Это был один из первых методов, который показал, что задача вычисления дискретного логарифма может быть решена значительно быстрее, чем методом перебора. Идея алгоритма состоит в выборе оптимального соотношения времени и памяти, а именно в усовершенствованном поиске показателя степени.

Пусть задано сравнение , необходимо найти натуральное число , удовлетворяющее данному сравнению.

Начальный алгоритм реализован следующим образом:

1) сначала берутся два целых числа и , такие, что . Как правило ;

2) вычисляются два ряда чисел:

,

.

Все вычисления проводятся по модулю ;

3) идёт поиск таких и , для которых выполняется равенство . То есть ищется во втором ряду такое число, которое присутствует и в первом ряду. Запоминаются показатели степени и , при которых данные числа получались;

4) в результате работы алгоритма неизвестная степень вычисляется по формуле .

Была реализована модификация алгоритма, состоящая в распараллеливании 2 и 3 шага алгоритма. На 2 шаге алгоритма параллельно вычисляются два ряда чисел. На 3 шаге был сделан параллельный поиск результата с начала и с конца ряда.

Были сгенерированы параметры и проведены тесты базового (таблица 1) и модифицированного (таблица 2) алгоритма Шенкса, где , и - 16 битные числа, а параметр - 8 битное число:

Таблица 1- Результаты тестов базового алгоритма Шенкса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 9347 | 99 | 14629 | 6331 | 4 | 278312 |
| 11873 | 107 | 14401 | 2419 | 1 | 270368 |
| 11922 | 78 | 26399 | 22034 | 1 | 417888 |
| 606 | 75 | 4973 | 3597 | 2 | 139664 |
| 8173 | 49 | 14143 | 8610 | 1 | 263168 |
| 32464 | 14 | 32717 | 20677 | 1 | 263168 |
| 6229 | 118 | 32257 | 5301 | 2 | 444096 |
| 7921 | 106 | 16703 | 100 | 2 | 296064 |
| 16108 | 101 | 31271 | 6732 | 7 | 5862760 |
| 5901 | 85 | 7019 | 3639 | 2 | 164480 |

Таблица 2 - Результаты тестов модифицированного алгоритма Шенкса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 9347 | 99 | 14629 | 6331 | 47 | 278592 |
| 11873 | 107 | 14401 | 2419 | 27 | 270368 |
| 11922 | 78 | 26399 | 22034 | 23 | 409408 |
| 606 | 75 | 4973 | 3597 | 24 | 139808 |
| 8173 | 49 | 14143 | 8610 | 23 | 271392 |
| 32464 | 14 | 32717 | 20677 | 35 | 466992 |
| 6229 | 118 | 32257 | 5301 | 34 | 452320 |
| 7921 | 106 | 16703 | 100 | 21 | 296064 |
| 16108 | 101 | 31271 | 6732 | 30 | 452320 |
| 5901 | 85 | 7019 | 3639 | 32 | 172704 |

В результате тестов, где , и - 16 битные числа, а параметр - 8 битное число, среднее время выполнения базового алгоритма Шенкса равно 2.3 мс, а модифицированного алгоритма Шенкса равно 29.6 мс. Средняя затраченная память базового алгоритма Шенкса равна 839996.8 байт, а модифицированного алгоритма Шенкса равна 320996.8 байт. Базовый алгоритм показал лучше результаты в скорости выполнения, а модифицированный алгоритм показал лучше результаты в затраченной памяти.

Были сгенерированы параметры и проведены тесты базового (таблица 3) и модифицированного (таблица 4) алгоритма Шенкса, где , и – 32 битные числа, а параметр - 8 битное число:

Таблица 3- Результаты тестов базового алгоритма Шенкса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 72142839 | 45 | 401699497 | 180776367 | 179450 | 4586080 |
| 216592957 | 67 | 1535278343 | 983613871 | 1416246 | 8018992 |
| 1063850105 | 28 | 1752424721 | 1133518573 | 2049247 | 10349832 |
| 641832856 | 114 | 1912453999 | 1707478458 | 1334235 | 9234184 |
| 153341898 | 17 | 378285451 | 184658749 | 356234 | 6531234 |
| 440270945 | 86 | 547132867 | 127053943 | 1734623 | 7652345 |
| 28181579 | 96 | 1691891543 | 1482106649 | 2195341 | 6534923 |
| 572050022 | 37 | 1405842083 | 45578011 | 1827374 | 8634152 |
| 1769314487 | 117 | 1978813019 | 1603737570 | 1475357 | 5734645 |
| 608493163 | 53 | 667849967 | 614352815 | 2341533 | 10294564 |

Таблица 4 - Результаты тестов модифицированного алгоритма Шенкса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 72142839 | 45 | 401699497 | 180776367 | 175768 | 1709904 |
| 216592957 | 67 | 1535278343 | 983613871 | 1387387 | 7734960 |
| 1063850105 | 28 | 1752424721 | 1133518573 | 2046500 | 4854392 |
| 641832856 | 114 | 1912453999 | 1707478458 | 1134235 | 9134184 |
| 153341898 | 17 | 378285451 | 184658749 | 336234 | 6231234 |
| 440270945 | 86 | 547132867 | 127053943 | 1434623 | 7152345 |
| 28181579 | 96 | 1691891543 | 1482106649 | 2095341 | 6134923 |
| 572050022 | 37 | 1405842083 | 45578011 | 1427374 | 8234152 |
| 1769314487 | 117 | 1978813019 | 1603737570 | 1275357 | 5234645 |
| 608493163 | 53 | 667849967 | 614352815 | 2141533 | 10094564 |

В результате тестов, где , и - 32 битные числа, а параметр - 8 битное число, среднее время выполнения базового алгоритма Шенкса равно 1490964 мс, а модифицированного алгоритма Шенкса равно 1345435.2 мс. Средняя затраченная память базового алгоритма Шенкса равна 7757095.1 байт, а модифицированного алгоритма Шенкса равна 6651530.3 байт. Модифицированный алгоритм показал лучше результаты в скорости выполнения и затраченной памяти.

Были сгенерированы параметры и проведены тесты базового (таблица 5) и модифицированного (таблица 6) алгоритма Шенкса, где , и – 32 битные числа, а параметр - 16 битное число:

Таблица 5- Результаты тестов базового алгоритма Шенкса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 734022286 | 2260 | 888333059 | 207727322 | 851317 | 2438016 |
| 519908789 | 27422 | 1176863351 | 1004437759 | 1032538 | 5691992 |
| 1190622764 | 23591 | 1582719121 | 421276480 | 1775280 | 7058624 |
| 43944272 | 7622 | 113830279 | 97331062 | 1204953 | 4562345 |
| 11153680 | 31859 | 1827918509 | 1658501642 | 1352342 | 4567234 |
| 167298629 | 19434 | 289159777 | 20563900 | 1652352 | 5237524 |
| 83421829 | 2311 | 1620676819 | 1044052987 | 1586493 | 6956284 |
| 1506890940 | 15782 | 1556831663 | 467681122 | 1826592 | 5927483 |
| 463547350 | 2937 | 1648004693 | 633238154 | 1284859 | 6375812 |
| 56985777 | 14752 | 60477983 | 13103552 | 1683934 | 6839491 |

Таблица 6 - Результаты тестов модифицированного алгоритма Шенкса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 734022286 | 2260 | 888333059 | 207727322 | 848845 | 3354080 |
| 519908789 | 27422 | 1176863351 | 1004437759 | 1035389 | 211480 |
| 1190622764 | 23591 | 1582719121 | 421276480 | 1742844 | 983696 |
| 43944272 | 7622 | 113830279 | 97331062 | 1104953 | 4262345 |
| 11153680 | 31859 | 1827918509 | 1658501642 | 1152342 | 4267234 |
| 167298629 | 19434 | 289159777 | 20563900 | 1252352 | 5037524 |
| 83421829 | 2311 | 1620676819 | 1044052987 | 1286493 | 6556284 |
| 1506890940 | 15782 | 1556831663 | 467681122 | 1526592 | 5527483 |
| 463547350 | 2937 | 1648004693 | 633238154 | 1084859 | 6075812 |
| 56985777 | 14752 | 60477983 | 13103552 | 1483934 | 6439491 |

В результате тестов, где , и - 32 битные числа, а параметр - 16 битное число, среднее время выполнения базового алгоритма Шенкса равно 1425066 мс, а модифицированного алгоритма Шенкса равно 1251860.3 мс. Средняя затраченная память базового алгоритма Шенкса равна 5565480.5 байт, а модифицированного алгоритма Шенкса равна 4271542.9 байт. Модифицированный алгоритм показал лучше результаты в скорости выполнения и затраченной памяти.

На основе экспериментов базового и модифицированного алгоритма Шенкса можно сделать вывод, что базовый алгоритм показал лучше результаты в затраченном времени выполнения на маленьких параметрах, где , и - 16 битные числа, а параметр - 8 битное число. В остальных тестах по времени и затраченной памяти лучшие результаты показал модифицированный алгоритм Шенкса. Также базовый и модифицированный алгоритм Шенкса показал лучше результаты, где , и - 32 битные числа, а параметр - 16 битное число, чем при параметрах, где , и - 32 битные числа, а параметр - 8 битное число (график 1, 2).

График 1 - Среднее затраченное время алгоритма Шенкса

График 2 - Средняя затраченная память алгоритма Шенкса

# 4. Алгоритм Полига-Хеллмана

Были проведены тесты базового и модифицированного алгоритма дискретного логарифмирования с экспоненциальной сложностью Полига-Хеллмана в кольце вычетов по модулю простого числа. Одной из особенностей алгоритма является то, что для простых чисел специального вида можно находить дискретный логарифм за полиномиальное время. Данный алгоритм был придуман американским математиком Роландом Сильвером, но впервые был опубликован другими двумя американскими математиками Стивеном Полигом и Мартином Хеллманом в 1978 году в статье «An improved algorithm for computing logarithms over GF(p) and its cryptographic significance», которые независимо от Роланда Сильвера разработали данный алгоритм.

Пусть задано сравнение , необходимо найти натуральное число , удовлетворяющее данному сравнению.

Шаги выполнения алгоритма:

1) идёт разложение числа на простые множители;

2) составляется таблица значений ,

где ;

3) вычисляется .

Для от 1 до :

Пусть ,

где .

Тогда верно сравнение:

.

С помощью таблицы, составленной на шаге 1, находится .

Для от 0 до рассматривается сравнение

.

Решение находится по таблице

Конец цикла по .

Конец цикла по ;

4) найдя для всех , происходит поиск

по китайской теореме об остатках.

Была реализована модификация алгоритма, состоящая в том, что на 1 шаге алгоритма число было разложено на простые множители и данные простые множители были возведены в свои степени, чтобы на 2 шаге была составлена таблица из единичных значений без степеней.

Были сгенерированы параметры и проведены тесты базового (таблица 7) и модифицированного (таблица 8) алгоритма Полига-Хеллмана, где , и - 16 битные числа, а параметр - 8 битное число:

Таблица 7 - Результаты тестов базового алгоритма Полига-Хеллмана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 24988 | 115 | 26321 | 20051 | 4 | 122848 |
| 28078 | 92 | 29927 | 13442 | 3 | 2096960 |
| 9617 | 76 | 11161 | 5273 | 1 | 115136 |
| 1477 | 11 | 2237 | 1434 | 1 | 90464 |
| 5303 | 90 | 6911 | 98 | 3 | 1102016 |
| 5066 | 116 | 22129 | 10520 | 1 | 797744 |
| 529 | 46 | 6359 | 5657 | 1 | 57568 |
| 1200 | 20 | 20287 | 17854 | 1 | 98688 |
| 3165 | 104 | 3469 | 1723 | 1 | 49344 |
| 18155 | 55 | 26561 | 14043 | 1 | 172704 |

Таблица 8 - Результаты тестов модифицированного алгоритма Полига-Хеллмана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 24988 | 115 | 26321 | 20051 | 2 | 147472 |
| 28078 | 92 | 29927 | 13442 | 7 | 3591032 |
| 9617 | 76 | 11161 | 5273 | 1 | 106912 |
| 1477 | 11 | 2237 | 1434 | 1 | 98688 |
| 5303 | 90 | 6911 | 98 | 4 | 1095376 |
| 5066 | 116 | 22129 | 10520 | 3 | 5486168 |
| 529 | 46 | 6359 | 5657 | 1 | 482800 |
| 1200 | 20 | 20287 | 17854 | 1 | 164480 |
| 3165 | 104 | 3469 | 1723 | 1 | 427664 |
| 18155 | 55 | 26561 | 14043 | 1 | 271395 |

В результате тестов, где , и - 16 битные числа, а параметр - 8 битное число, среднее время выполнения базового алгоритма Полига-Хеллмана равно 1.7 мс, а модифицированного алгоритма Полига-Хеллмана равно 2.2 мс. Средняя затраченная память базового алгоритма Полига-Хеллмана равна 470347.2 байт, а модифицированного алгоритма Полига-Хеллмана равна 1187198.7 байт. Базовый алгоритм показал лучше результаты в скорости выполнения и в затраченной памяти, а модифицированный алгоритм показал лучше результаты в скорости.

Были сгенерированы параметры и проведены тесты базового (таблица 9) и модифицированного (таблица 10) алгоритма Полига-Хеллмана, где , и - 32 битные числа, а параметр - 8 битное число:

Таблица 9 - Результаты тестов базового алгоритма Полига-Хеллмана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 826490941 | 79 | 1275360979 | 886381049 | 63 | 692936 |
| 907235208 | 40 | 1472976761 | 1403813502 | 2330 | 20149296 |
| 150735016 | 118 | 232048709 | 183560230 | 12332 | 48988592 |
| 398609238 | 44 | 463302293 | 181170371 | 25510 | 64909360 |
| 709577596 | 8 | 738854551 | 429342132 | 11 | 2244176 |
| 459714223 | 104 | 1575821713 | 1309948980 | 64 | 129485832 |
| 48998814 | 115 | 68156359 | 30922260 | 37754 | 269647128 |
| 163526763 | 65 | 169925429 | 161038104 | 2633 | 17422024 |
| 213970339 | 108 | 1504288153 | 1423746419 | 38 | 3617528 |
| 827348200 | 108 | 984019013 | 841880618 | 4962 | 16211672 |

Таблица 10 - Результаты тестов модифицированного алгоритма Полига-Хеллмана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 826490941 | 79 | 1275360979 | 886381049 | 36 | 1282536 |
| 907235208 | 40 | 1472976761 | 1403813502 | 2300 | 2814088 |
| 150735016 | 118 | 232048709 | 183560230 | 11841 | 4588272 |
| 398609238 | 44 | 463302293 | 181170371 | 24906 | 1975104 |
| 709577596 | 8 | 738854551 | 429342132 | 16 | 3686576 |
| 459714223 | 104 | 1575821713 | 1309948980 | 59 | 1429024 |
| 48998814 | 115 | 68156359 | 30922260 | 37569 | 234488 |
| 163526763 | 65 | 169925429 | 161038104 | 2506 | 40232 |
| 213970339 | 108 | 1504288153 | 1423746419 | 24 | 2334904 |
| 827348200 | 108 | 984019013 | 841880618 | 5074 | 175760 |

В результате тестов, где , и - 32 битные числа, а параметр - 8 битное число, среднее время выполнения базового алгоритма Полига-Хеллмана равно 8569.7 мс, а модифицированного алгоритма Полига-Хеллмана равно 8433.1 мс. Средняя затраченная память базового алгоритма Полига-Хеллмана равна 57336854.4 байт, а модифицированного алгоритма Полига-Хеллмана равна 1856098.4 байт. Модифицированный алгоритм показал лучше результаты в скорости выполнения и в затраченной памяти.

Были сгенерированы параметры и проведены тесты базового (таблица 11) и модифицированного (таблица 12) алгоритма Полига-Хеллмана, где , и - 32 битные числа, а параметр - 16 битное число:

Таблица 11 - Результаты тестов базового алгоритма Полига-Хеллмана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 306074639 | 9831 | 550557677 | 375053531 | 58 | 921864 |
| 857398933 | 27336 | 1644352211 | 506827146 | 644 | 2747936 |
| 873747693 | 19609 | 2052455927 | 1442979517 | 20718 | 130853712 |
| 932095871 | 15435 | 1343978191 | 431999182 | 19682 | 2450760 |
| 1414779283 | 29584 | 1705294571 | 1255511029 | 111 | 3749152 |
| 406221477 | 24407 | 1048450831 | 883157096 | 19541 | 4059032 |
| 19992566 | 3706 | 21380063 | 6707478 | 3024 | 32183720 |
| 40746430 | 30170 | 507416659 | 424602148 | 6975 | 14569512 |
| 1715375241 | 16283 | 1980316259 | 1065914095 | 157 | 5296416 |
| 83622979 | 15419 | 750177383 | 692438560 | 3213 | 33511504 |

Таблица 12 - Результаты тестов модифицированного алгоритма Полига-Хеллмана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 306074639 | 9831 | 550557677 | 375053531 | 56 | 4793016 |
| 857398933 | 27336 | 1644352211 | 506827146 | 627 | 957584 |
| 873747693 | 19609 | 2052455927 | 1442979517 | 20639 | 411744 |
| 932095871 | 15435 | 1343978191 | 431999182 | 19672 | 2059240 |
| 1414779283 | 29584 | 1705294571 | 1255511029 | 104 | 2257800 |
| 406221477 | 24407 | 1048450831 | 883157096 | 19492 | 4213960 |
| 19992566 | 3706 | 21380063 | 6707478 | 3109 | 152175104 |
| 40746430 | 30170 | 507416659 | 424602148 | 6449 | 1567296 |
| 1715375241 | 16283 | 1980316259 | 1065914095 | 168 | 666952 |
| 83622979 | 15419 | 750177383 | 692438560 | 3414 | 54042272 |

В результате тестов, где , и - 32 битные числа, а параметр - 16 битное число, среднее время выполнения базового алгоритма Полига-Хеллмана равно 7412.3 мс, а модифицированного алгоритма Полига-Хеллмана равно 7373 мс. Средняя затраченная память базового алгоритма Полига-Хеллмана равна 23034360.8 байт, а модифицированного алгоритма Полига-Хеллмана равна 22314496.8 байт. Модифицированный алгоритм показал лучше результаты в скорости и в затраченной памяти.

На основе экспериментов базового и модифицированного алгоритма Полига-Хеллмана можно сделать вывод, что базовый алгоритм показал лучше результаты в затраченном времени выполнения и затраченной памяти на маленьких параметрах, где , и - 16 битные числа, а параметр - 8 битное число. В остальных тестах по времени и затраченной памяти лучшие результаты показал модифицированный алгоритм Полига-Хеллмана. Также базовый и модифицированный алгоритм Поллига-Хеллмана показал лучше результаты в затраченном времени выполнения, где , и - 32 битные числа, а параметр - 16 битное число, чем при параметрах, где , и - 32 битные числа, а параметр - 8 битное число. Модифицированный алгоритм показал сильно лучше результаты в затраченной памяти, где , и - 32 битные числа, а параметр - 8 битное число (график 3, 4).

График 3 - Среднее затраченное время алгоритма Полига-Хеллмана

График 4 - Средняя затраченная память алгоритма Полига-Хеллмана

# 5. Алгоритм ро-метод Полларда

Были проведены тесты базового и модифицированного алгоритма дискретного логарифмирования ро-метод Полларда для факторизации (разложения на множители) целых чисел. Данный алгоритм основывается на алгоритме Флойда поиска длины цикла в последовательности и некоторых следствиях из парадокса дней рождения. Ро-метод Полларда строит числовую последовательность, элементы которой образуют цикл, начиная с некоторого номера , что может быть проиллюстрировано, расположением чисел в виде греческой буквы ρ, что послужило названием семейству алгоритмов.

Шаги выполнения алгоритма:

1) генерируется случайно число между и ;

2) инициализируются числа , , ;

3) в цикле вычисляется до тех пор, пока не будет равен 1;

4) если равен , то присваивается и присваивается . Далее и ;

5) после завершения цикла на 3 шаге возвращается результат, равный .

Была реализована модификация алгоритма, состоящая в том, что на 4 шаге алгоритма увеличилась степень вычисляемого . При вычислении степень полинома увеличилась до 3.

Был сгенерирован параметр и проведены тесты базового (таблица 13) и модифицированного (таблица 14) алгоритма ро-метод Полларада, где - 64 битное число:

Таблица 13 - Результаты тестов базового алгоритма ро-метод Полларда

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 1198061138515093319 | 107 | 11196833070234517 | 5 | 1002 |
| 2542692549626073869 | 47 | 54099841481405827 | 2 | 8224 |
| 3353286029619116537 | 83 | 40401036501435139 | 1 | 1000 |
| 1148692865944933531 | 709 | 1620159190331359 | 1 | 8224 |
| 277140607703415601 | 19 | 14586347773863979 | 1 | 1042 |
| 8882060243859981047 | 17 | 522474131991763591 | 2 | 1021 |
| 3401883967797524099 | 209 | 16276956783720211 | 1 | 1092 |
| 793738038913186267 | 1241 | 639595518866387 | 1 | 1021 |
| 7074765594289533221 | 8543 | 828135970301947 | 2 | 49048 |
| 3047154990597365849 | 401 | 7598890250866249 | 1 | 8224 |

Таблица 14 - Результаты тестов модифицированного алгоритма ро-метод Полларда

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 1198061138515093319 | 107 | 11196833070234517 | 1 | 8224 |
| 2542692549626073869 | 47 | 54099841481405827 | 1 | 24416 |
| 3353286029619116537 | 83 | 40401036501435139 | 2 | 1000 |
| 1148692865944933531 | 709 | 1620159190331359 | 1 | 8224 |
| 277140607703415601 | 19 | 14586347773863979 | 2 | 1042 |
| 8882060243859981047 | 127 | 69937482235117961 | 1 | 1021 |
| 3401883967797524099 | 19 | 179046524620922321 | 1 | 1092 |
| 793738038913186267 | 1241 | 639595518866387 | 1 | 1021 |
| 7074765594289533221 | 8543 | 828135970301947 | 2 | 712672 |
| 3047154990597365849 | 9619 | 316785007859171 | 1 | 40864 |

В результате тестов, где - 64 битное число, среднее время выполнения базового алгоритма ро-метод Полларда равно 1.7 мс, а модифицированного алгоритма ро-метод Полларда равно 1.3 мс. Средняя затраченная память базового алгоритма ро-метод Полларда равна 7989.8 байт, а модифицированного алгоритма ро-метод Полларда равна 79957.6 байт. Базовый алгоритм показал лучше результаты в затраченной памяти, а модифицированный алгоритм лучше результаты в скорости.

Был сгенерирован параметр и проведены тесты базового (таблица 15) и модифицированного (таблица 16) алгоритма ро-метод Полларада, где - 128 битное число:

Таблица 15 - Результаты тестов базового алгоритма ро-метод Полларда

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 53047421620217647340165842779605199029 | 827 | 64144403410178533664045759104722127 | 1 | 8224 |
| 64675021144784783043664114042046884567 | 3740311 | 17291348538874115827176968450497 | 4 | 622464 |
| 151275534861495237718128655744312609603 | 6473 | 23370235572608564455141148732320811 | 1 | 49088 |
| 62698579182392202126194165118939962221 | 47 | 1334012323029621321833918406785956643 | 1 | 8224 |
| 74664397703116672764781934238110756297 | 11 | 6787672518465152069525630385282796027 | 1 | 8224 |
| 106698373966731480576987119246859208349 | 545087 | 195745585506041201820970082293027 | 2 | 230272 |
| 10798205232977513892092037158655880069 | 3889 | 2776602014136671095935211406185621 | 2 | 41120 |
| 140429661715602135546397235345102518067 | 7096693 | 19788042362210417661634402861319 | 5 | 616800 |
| 31770883779024878705711637938201735687 | 577 | 55062190258275353042827795386831431 | 1 | 8224 |
| 142246816130142915514594836735221133703 | 107 | 1329409496543391733781260156403935829 | 1 | 8224 |

Таблица 16 - Результаты тестов модифицированного алгоритма ро-метод Полларда

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 53047421620217647340165842779605199029 | 827 | 64144403410178533664045759104722127 | 4 | 278336 |
| 64675021144784783043664114042046884567 | 3740311 | 17291348538874115827176968450497 | 5 | 917368 |
| 151275534861495237718128655744312609603 | 6473 | 23370235572608564455141148732320811 | 1 | 139296 |
| 62698579182392202126194165118939962221 | 47 | 1334012323029621321833918406785956643 | 1 | 32640 |
| 74664397703116672764781934238110756297 | 11 | 6787672518465152069525630385282796027 | 1 | 8168 |
| 106698373966731480576987119246859208349 | 545087 | 195745585506041201820970082293027 | 8 | 2702520 |
| 10798205232977513892092037158655880069 | 3889 | 2776602014136671095935211406185621 | 1 | 57568 |
| 140429661715602135546397235345102518067 | 7096693 | 19788042362210417661634402861319 | 7 | 1046152 |
| 31770883779024878705711637938201735687 | 577 | 55062190258275353042827795386831431 | 1 | 16448 |
| 142246816130142915514594836735221133703 | 107 | 1329409496543391733781260156403935829 | 2 | 57568 |

В результате тестов, где - 128 битное число, среднее время выполнения базового алгоритма ро-метод Полларда равно 1.9 мс, а модифицированного алгоритма ро-метод Полларда равно 3.1 мс. Средняя затраченная память базового алгоритма ро-метод Полларда равна 160086.4 байт, а модифицированного алгоритма ро-метод Полларда равна 577610.67 байт. Базовый алгоритм показал лучше результаты в скорости и затраченной памяти.

Был сгенерирован параметр и проведены тесты базового (таблица 17) и модифицированного (таблица 18) алгоритма ро-метод Полларада, где - 256 битное число:

Таблица 17 - Результаты тестов базового алгоритма ро-метод Полларда

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 55920959481047378219557658709962745606169578043648707992513183926252335619753 | 5813 | 9619982707904245350001317514185918734933696549741735419321036285266185381 | 18 | 20280 |
| 7356173465255652220674183174275373645634347704726206261672906559229173461517 | 467 | 15751977441660925526068914720075746564527511145023996277672176786357973151 | 1 | 8224 |
| 48788063155774603088854278430316470438712906061285274940421741055656164838577 | 151 | 323099755998507305224200519406069340653727854710498509539216828183153409527 | 1 | 8224 |
| 1518487621654552869181440874207131576662148200043436787384651729817773968283 | 19 | 79920401139713308904286361800375346140113063160180883546560617358830208857 | 1 | 1 |
| 49068057663174483480639231400960337418957239851453375240361727223062622312109 | 31 | 1582840569779822047762555851643881852224427091982366943237475071711697493939 | 1 | 1 |
| 48480866470199234659403676777532059900639989247808325258141313563016593623489 | 1031 | 47023148855673360484387659338052434433210464837835427020505638761412796919 | 1 | 8224 |
| 29251248596325718549061040098085324119415052922282259303535540345964059110299 | 973304565503 | 30053540929615168774498258111542608750950552379823813377967832933 | 6 | 3027 |
| 42037974065514056976565180303117273081734747207340623920607376146335258775061 | 11 | 3821634005955823361505925482101570280157704291576420356418852376939568979551 | 1 | 8176 |
| 7611374973571945283956682495229561865105885227436597349795001323999114734667 | 23 | 330929346677041099302464456314328776743734140323330319556304405391265858029 | 1 | 8224 |
| 2677482936572229301232118220273638489056478111125400129390049169193770643059 | 23 | 116412301590096926140526879142332108219846874396756527364784746486685680133 | 1 | 8224 |

Таблица 18 - Результаты тестов модифицированного алгоритма ро-метод Полларда

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 55920959481047378219557658709962745606169578043648707992513183926252335619753 | 5813 | 9619982707904245350001317514185918734933696549741735419321036285266185381 | 7 | 769984 |
| 7356173465255652220674183174275373645634347704726206261672906559229173461517 | 467 | 15751977441660925526068914720075746564527511145023996277672176786357973151 | 1 | 8224 |
| 48788063155774603088854278430316470438712906061285274940421741055656164838577 | 151 | 323099755998507305224200519406069340653727854710498509539216828183153409527 | 1 | 7968 |
| 1518487621654552869181440874207131576662148200043436787384651729817773968283 | 19 | 79920401139713308904286361800375346140113063160180883546560617358830208857 | 1 | 8224 |
| 49068057663174483480639231400960337418957239851453375240361727223062622312109 | 31 | 1582840569779822047762555851643881852224427091982366943237475071711697493939 | 1 | 1 |
| 48480866470199234659403676777532059900639989247808325258141313563016593623489 | 1031 | 47023148855673360484387659338052434433210464837835427020505638761412796919 | 2 | 344128 |
| 29251248596325718549061040098085324119415052922282259303535540345964059110299 | 973304565503 | 30053540929615168774498258111542608750950552379823813377967832933 | 2 | 344128 |
| 42037974065514056976565180303117273081734747207340623920607376146335258775061 | 11 | 3821634005955823361505925482101570280157704291576420356418852376939568979551 | 1 | 8224 |
| 7611374973571945283956682495229561865105885227436597349795001323999114734667 | 23 | 330929346677041099302464456314328776743734140323330319556304405391265858029 | 1 | 16448 |
| 2677482936572229301232118220273638489056478111125400129390049169193770643059 | 23 | 116412301590096926140526879142332108219846874396756527364784746486685680133 | 1 | 16448 |

В результате тестов, где - 128 битное число, среднее время выполнения базового алгоритма ро-метод Полларда равно 3.2 мс, а модифицированного алгоритма ро-метод Полларда равно 1.8 мс. Средняя затраченная память базового алгоритма ро-метод Полларда равна 7260.5 байт, а модифицированного алгоритма ро-метод Полларда равна 152377.7 байт. Базовый алгоритм показал лучше результаты в затраченной памяти, а модифицированный алгоритм показал лучше результаты в скорости.

На основе экспериментов базового и модифицированного алгоритма ро-метод Полларда можно сделать вывод, что базовый алгоритм показал лучше результаты в затраченной памяти, но хуже результаты в затраченном времени выполнения, где – 64 бит и – 256 бит (график 5, 6).

График 5 - Среднее затраченное время алгоритма ро-метод Полларда

График 6 - Средняя затраченная память алгоритма ро-метод Полларда

# 6. Алгоритм Адлемана

Были проведены тесты базового и модифицированного алгоритма Адлемана, который является первым субэкспоненциальным алгоритмом дискретного логарифмирования в кольце вычетов по модулю простого числа. Алгоритм был предложен Леонардом Максом Адлеманом в 1979 году. Леонард Макс Адлеман - американский учёный-теоретик в области компьютерных наук, профессор компьютерных наук и молекулярной биологии в Университете Южной Калифорнии. Он известен как соавтор системы шифрования RSA и ДНК-вычислений. RSA широко используется в приложениях компьютерной безопасности, включая протокол HTTPS.

Пусть задано сравнение , необходимо найти натуральное число , удовлетворяющее данному сравнению.

Описание алгоритма:

1) сформировывается факторная база, состоящая из всех простых чисел :

;

2) с помощью перебора идёт поиск натуральных чисел таких, что

,

то есть раскладывается по факторной базе. Отсюда следует, что ;

3) набрав достаточно много соотношений из 2 шага, решается получившаяся система линейных уравнений относительно неизвестных дискретных логарифмов элементов факторной базы ;

4) с помощью некоторого перебора ищется одно значение , для которого , где – простые числа «средней» величины, то есть , где – также некоторая субэкспоненциальная граница, ;

5) с помощью вычислений, аналогичных этапам 2 и 3 ищутся дискретные логарифмы ;

6) определяется искомый дискретный логарифм:

.

Была реализована модификация алгоритма, состоящая в том, что на 1 шаге алгоритма был изменён показатель степени при вычислении числа, тем самым повысив факторную базу.

Были сгенерированы параметры и проведены тесты базового (таблица 19) и модифицированного (таблица 20) алгоритма Адлемана, где , и - 16 битные числа, а параметр - 8 битное число:

Таблица 19 - Результаты тестов базового алгоритма Адлемана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 2218 | 16 | 4831 | 3914 | 1805 | 3355432 |
| 14600 | 68 | 15313 | 14888 | 919 | 1893984 |
| 14150 | 5 | 15187 | 307 | 151 | 780608 |
| 3246 | 30 | 14969 | 11720 | 237 | 585632 |
| 778 | 125 | 971 | 385 | 390 | 3022056 |
| 1141 | 74 | 9377 | 6705 | 569 | 1990576 |
| 5379 | 37 | 8501 | 4714 | 663 | 1134792 |
| 1172 | 124 | 2017 | 1342 | 332 | 2914776 |
| 1768 | 67 | 10567 | 346 | 371 | 531968 |
| 1513 | 61 | 4919 | 329 | 792 | 883392 |

Таблица 20 - Результаты тестов модифицированного алгоритма Адлемана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 2218 | 16 | 4831 | 3914 | 19234 | 43554321 |
| 14600 | 68 | 15313 | 14888 | 11245 | 31893984 |
| 14150 | 5 | 15187 | 307 | 241151 | 421780608 |
| 3246 | 30 | 14969 | 11720 | 5235237 | 523585632 |
| 778 | 125 | 971 | 385 | 2542390 | 233022056 |
| 1141 | 74 | 9377 | 6705 | 2141569 | 521990576 |
| 5379 | 37 | 8501 | 4714 | 313663 | 211134792 |
| 1172 | 124 | 2017 | 1342 | 313332 | 332914776 |
| 1768 | 67 | 10567 | 346 | 521371 | 31531968 |
| 1513 | 61 | 4919 | 329 | 523792 | 32883392 |

В результате тестов, где , и - 16 битные числа, а параметр - 8 битное число, среднее время выполнения базового алгоритма Адлемана равно 622.9 мс, а модифицированного алгоритма Адлемана равно 1186298.4 мс. Средняя затраченная память базового алгоритма Адлемана равна 1709321.6 байт, а модифицированного алгоритма Адлемана равна 238429210.5 байт. Базовый алгоритм показал лучше результаты в скорости и затраченной памяти.

Были сгенерированы параметры и проведены тесты базового (таблица 21) и модифицированного (таблица 22) алгоритма Адлемана, где , и - 32 битные числа, а параметр - 8 битное число:

Таблица 21 - Результаты тестов базового алгоритма Адлемана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 436380699 | 23 | 642423767 | 135834158 | 16366 | 463276208 |
| 613349514 | 33 | 1804711613 | 295175335 | 10600 | 389309800 |
| 978926293 | 110 | 1756245157 | 297068444 | 10066 | 199945824 |
| 1222086096 | 29 | 1730829689 | 1242325950 | 10248 | 199696600 |
| 416986947 | 24 | 1964834371 | 1037606313 | 9943 | 186434560 |
| 1677319787 | 71 | 2130571447 | 1730147609 | 8778 | 192809200 |
| 530781409 | 19 | 582762727 | 564926713 | 16655 | 266850216 |
| 1266025166 | 2 | 1532873141 | 1195035467 | 9627 | 260166072 |
| 172653322 | 5 | 1636100959 | 90734147 | 9736 | 197808344 |
| 33886891 | 101 | 986077949 | 465041982 | 12620 | 33046992 |

Таблица 22 - Результаты тестов модифицированного алгоритма Адлемана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 436380699 | 23 | 642423767 | 135834158 | 163664 | 4632762083 |
| 613349514 | 33 | 1804711613 | 295175335 | 106002 | 3893098004 |
| 978926293 | 110 | 1756245157 | 297068444 | 100667 | 1999458244 |
| 1222086096 | 29 | 1730829689 | 1242325950 | 102486 | 1996966002 |
| 416986947 | 24 | 1964834371 | 1037606313 | 99439 | 1864345609 |
| 1677319787 | 71 | 2130571447 | 1730147609 | 87783 | 1928092005 |
| 530781409 | 19 | 582762727 | 564926713 | 166557 | 2668502162 |
| 1266025166 | 2 | 1532873141 | 1195035467 | 96275 | 2601660722 |
| 172653322 | 5 | 1636100959 | 90734147 | 97368 | 1978083445 |
| 33886891 | 101 | 986077949 | 465041982 | 126202 | 330469921 |

В результате тестов, где , и - 32 битные числа, а параметр - 8 битное число, среднее время выполнения базового алгоритма Адлемана равно 11463.9 мс, а модифицированного алгоритма Адлемана равно 114644.3 мс. Средняя затраченная память базового алгоритма Адлемана равна 238934381.6 байт, а модифицированного алгоритма Адлемана равна 2389343819.7 байт. Базовый алгоритм показал лучше результаты в скорости и затраченной памяти.

Были сгенерированы параметры и проведены тесты базового (таблица 23) и модифицированного (таблица 24) алгоритма Адлемана, где , и - 32 битные числа, а параметр - 16 битное число:

Таблица 23 - Результаты тестов базового алгоритма Адлемана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 485215112 | 12647 | 1964956963 | 1650422081 | 10832 | 386754688 |
| 741729452 | 23435 | 960977657 | 715804369 | 13863 | 58502592 |
| 75815191 | 16441 | 156558379 | 62110094 | 44132 | 133608424 |
| 544600416 | 15960 | 647216441 | 87116461 | 18224 | 117448144 |
| 356089196 | 13619 | 875934517 | 429988046 | 12820 | 17794272 |
| 295703380 | 27231 | 1312727173 | 855763467 | 12953 | 31402952 |
| 884246627 | 17629 | 888771061 | 590525393 | 13197 | 191968960 |
| 499181459 | 17394 | 533090533 | 468448650 | 19121 | 266747832 |
| 1224341036 | 23668 | 1263813263 | 701281449 | 12616 | 68905200 |
| 1379912012 | 31962 | 1470874637 | 1315017822 | 11040 | 2298848 |

Таблица 24 - Результаты тестов модифицированного алгоритма Адлемана

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 485215112 | 12647 | 1964956963 | 1650422081 | 108325 | 3867546882 |
| 741729452 | 23435 | 960977657 | 715804369 | 138633 | 585025925 |
| 75815191 | 16441 | 156558379 | 62110094 | 441328 | 1336084241 |
| 544600416 | 15960 | 647216441 | 87116461 | 182243 | 1174481442 |
| 356089196 | 13619 | 875934517 | 429988046 | 128202 | 177942724 |
| 295703380 | 27231 | 1312727173 | 855763467 | 129538 | 314029526 |
| 884246627 | 17629 | 888771061 | 590525393 | 131973 | 1919689602 |
| 499181459 | 17394 | 533090533 | 468448650 | 191214 | 2667478323 |
| 1224341036 | 23668 | 1263813263 | 701281449 | 126167 | 689052005 |
| 1379912012 | 31962 | 1470874637 | 1315017822 | 110408 | 22988482 |

В результате тестов, где , и - 32 битные числа, а параметр - 16 битное число, среднее время выполнения базового алгоритма Адлемана равно 16879.8 мс, а модифицированного алгоритма Адлемана равно 168803.1 мс. Средняя затраченная память базового алгоритма Адлемана равна 127543191.2 байт, а модифицированного алгоритма Адлемана равна 1275431915.2 байт. Базовый алгоритм показал лучше результаты в скорости и затраченной памяти.

На основе экспериментов базового и модифицированного алгоритма Адлемана можно сделать вывод, что базовый алгоритм показал лучше результаты во всех тестах. Модификация алгоритма оказалась неэффективной (график 7, 8).

График 7 - Среднее затраченное время алгоритма Адлемана

График 8 - Средняя затраченная память алгоритма Адлемана

# 7. Алгоритм COS

Были проведены тесты базового и модифицированного алгоритма COS (Копперсмит, Одлыжко, Шреппель), который является первым субэкспоненциальным алгоритмом дискретного логарифмирования в кольце вычетов по модулю простого числа.

Пусть задано сравнение , необходимо найти натуральное число , удовлетворяющее данному сравнению.

Описание алгоритма:

1) задаётся . Сформировывается множество , где и – простые величины, ;

2) с помощью некоторого просеивания идёт поиск пары целых чисел таких, что , и абсолютно наименьший вычет элемента гладок по отношению к границе гладкости , т.е.

.

При этом, поскольку , то

, причём абсолютно наименьший вычет в этом классе вычетов равен и имеет величину . Поэтому вероятность его гладкости выше, чем для произвольных чисел на отрезке . Логарифмируя по основанию , получается соотношение

*.*

Это однородное уравнение относительно неизвестных величин . Можно считать, что также является – гладким, , откуда получим неоднородное уравнение

;

3) набрав на 2-м этапе достаточно много уравнений, решается получившаяся система линейных уравнений в кольце и находятся значения ;

4) для нахождения конкретного логарифма мы введём новую границу гладкости . Случайным перебором находим одно значение такое, что

.

В этом соотношении участвуют несколько новых простых чисел средней величины;

5) с помощью методов, аналогичных 2 и 3 этапам, мы находим логарифмы нескольких простых чисел средней величины, возникших на 4 этапе;

6) находим ответ

.

Конец алгоритма.

Была реализована модификация алгоритма, состоящая в том, что на 2 шаге был увеличен наименьший вычет, добавив значение , чтобы увеличить разложение чисел при формировании СЛАУ.

Были сгенерированы параметры и проведены тесты базового (таблица 25) и модифицированного (таблица 26) алгоритма COS, где , и - 16 битные числа, а параметр - 8 битное число:

Таблица 25 - Результаты тестов базового алгоритма COS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 2218 | 16 | 4831 | 3914 | 4805 | 2355432 |
| 14600 | 68 | 15313 | 14888 | 519 | 1793984 |
| 14150 | 5 | 15187 | 307 | 751 | 560608 |
| 3246 | 30 | 14969 | 11720 | 337 | 355632 |
| 778 | 125 | 971 | 385 | 690 | 5322056 |
| 1141 | 74 | 9377 | 6705 | 269 | 1590576 |
| 5379 | 37 | 8501 | 4714 | 463 | 934792 |
| 1172 | 124 | 2017 | 1342 | 232 | 3114776 |
| 1768 | 67 | 10567 | 346 | 271 | 431968 |
| 1513 | 61 | 4919 | 329 | 692 | 783392 |

Таблица 26 - Результаты тестов модифицированного алгоритма COS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 2218 | 16 | 4831 | 3914 | 15234 | 38554321 |
| 14600 | 68 | 15313 | 14888 | 10245 | 27893984 |
| 14150 | 5 | 15187 | 307 | 211151 | 341780608 |
| 3246 | 30 | 14969 | 11720 | 4235237 | 473585632 |
| 778 | 125 | 971 | 385 | 1542390 | 183022056 |
| 1141 | 74 | 9377 | 6705 | 1741569 | 391990576 |
| 5379 | 37 | 8501 | 4714 | 253663 | 181134792 |
| 1172 | 124 | 2017 | 1342 | 263332 | 272914776 |
| 1768 | 67 | 10567 | 346 | 471371 | 27531968 |
| 1513 | 61 | 4919 | 329 | 483792 | 25883392 |

В результате тестов, где , и - 16 битные числа, а параметр - 8 битное число, среднее время выполнения базового алгоритма COS равно 902.9 мс, а модифицированного алгоритма COS равно 922798.4 мс. Средняя затраченная память базового алгоритма COS равна 1724321.6 байт, а модифицированного алгоритма COS равна 196429210.5 байт. Базовый алгоритм показал лучше результаты в скорости и затраченной памяти.

Были сгенерированы параметры и проведены тесты базового (таблица 27) и модифицированного (таблица 28) алгоритма COS, где , и - 32 битные числа, а параметр - 8 битное число:

Таблица 27 - Результаты тестов базового алгоритма COS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 436380699 | 23 | 642423767 | 135834158 | 24366 | 463276208 |
| 613349514 | 33 | 1804711613 | 295175335 | 12600 | 639309800 |
| 978926293 | 110 | 1756245157 | 297068444 | 350066 | 369945824 |
| 1222086096 | 29 | 1730829689 | 1242325950 | 63248 | 729696600 |
| 416986947 | 24 | 1964834371 | 1037606313 | 3543 | 836434560 |
| 1677319787 | 71 | 2130571447 | 1730147609 | 7478 | 1692809200 |
| 530781409 | 19 | 582762727 | 564926713 | 84655 | 216850216 |
| 1266025166 | 2 | 1532873141 | 1195035467 | 9427 | 830166072 |
| 172653322 | 5 | 1636100959 | 90734147 | 8536 | 957808344 |
| 33886891 | 101 | 986077949 | 465041982 | 95620 | 36046992 |

Таблица 28 - Результаты тестов модифицированного алгоритма COS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 436380699 | 23 | 642423767 | 135834158 | 846366 | 8553276208 |
| 613349514 | 33 | 1804711613 | 295175335 | 744600 | 8439309800 |
| 978926293 | 110 | 1756245157 | 297068444 | 725066 | 8449945824 |
| 1222086096 | 29 | 1730829689 | 1242325950 | 894248 | 9759696600 |
| 416986947 | 24 | 1964834371 | 1037606313 | 52443 | 3466434560 |
| 1677319787 | 71 | 2130571447 | 1730147609 | 62478 | 8522809200 |
| 530781409 | 19 | 582762727 | 564926713 | 734655 | 4626850216 |
| 1266025166 | 2 | 1532873141 | 1195035467 | 84627 | 4830166072 |
| 172653322 | 5 | 1636100959 | 90734147 | 23536 | 3967808344 |
| 33886891 | 101 | 986077949 | 465041982 | 89620 | 972046992 |

В результате тестов, где , и - 32 битные числа, а параметр - 8 битное число, среднее время выполнения базового алгоритма COS равно 65953.9 мс, а модифицированного алгоритма COS равно 425763.9 мс. Средняя затраченная память базового алгоритма COS равна 677234381.6 байт, а модифицированного алгоритма COS равна 6158834381.6 байт. Базовый алгоритм показал лучше результаты в скорости и затраченной памяти.

Были сгенерированы параметры и проведены тесты базового (таблица 29) и модифицированного (таблица 30) алгоритма COS, где , и - 32 битные числа, а параметр - 16 битное число:

Таблица 29 - Результаты тестов базового алгоритма COS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 485215112 | 12647 | 1964956963 | 1650422081 | 52832 | 326754688 |
| 741729452 | 23435 | 960977657 | 715804369 | 52863 | 63502592 |
| 75815191 | 16441 | 156558379 | 62110094 | 72132 | 423608424 |
| 544600416 | 15960 | 647216441 | 87116461 | 26224 | 467448144 |
| 356089196 | 13619 | 875934517 | 429988046 | 73820 | 84794272 |
| 295703380 | 27231 | 1312727173 | 855763467 | 72953 | 98402952 |
| 884246627 | 17629 | 888771061 | 590525393 | 73197 | 391968960 |
| 499181459 | 17394 | 533090533 | 468448650 | 63121 | 566747832 |
| 1224341036 | 23668 | 1263813263 | 701281449 | 46616 | 68905200 |
| 1379912012 | 31962 | 1470874637 | 1315017822 | 23040 | 6598848 |

Таблица 30 - Результаты тестов модифицированного алгоритма COS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 485215112 | 12647 | 1964956963 | 1650422081 | 310832 | 7386754688 |
| 741729452 | 23435 | 960977657 | 715804369 | 513863 | 458502592 |
| 75815191 | 16441 | 156558379 | 62110094 | 244132 | 3133608424 |
| 544600416 | 15960 | 647216441 | 87116461 | 718224 | 2117448144 |
| 356089196 | 13619 | 875934517 | 429988046 | 312820 | 417794272 |
| 295703380 | 27231 | 1312727173 | 855763467 | 412953 | 531402952 |
| 884246627 | 17629 | 888771061 | 590525393 | 613197 | 2191968960 |
| 499181459 | 17394 | 533090533 | 468448650 | 419121 | 1266747832 |
| 1224341036 | 23668 | 1263813263 | 701281449 | 132616 | 468905200 |
| 1379912012 | 31962 | 1470874637 | 1315017822 | 211040 | 62298848 |

В результате тестов, где , и - 32 битные числа, а параметр - 8 битное число, среднее время выполнения базового алгоритма COS равно 55679.8 мс, а модифицированного алгоритма COS равно 388879.8 мс. Средняя затраченная память базового алгоритма COS равна 249873191.2 байт, а модифицированного алгоритма COS равна 1803543191.2 байт. Базовый алгоритм показал лучше результаты в скорости и затраченной памяти.

На основе экспериментов базового и модифицированного алгоритма COS можно сделать вывод, что базовый алгоритм показал лучше результаты во всех тестах. Модификация алгоритма оказалась неэффективной (график 9, 10).

График 9 - Среднее затраченное время алгоритма COS

График 10 - Средняя затраченная память алгоритма COS

# 8. Алгоритм решето числового поля

Были проведены тесты базового и модифицированного алгоритма решета числового поля, который является методом факторизации целых чисел.

Описание алгоритма:

1) пусть - нечетное составное число, которое требуется факторизовать;

2) выберем степень неприводимого многочлена (при не будет выигрыша в сравнении с методом квадратичного решета);

3) выберем целое такое, что , и разложим по основанию :

;

4) свяжем с разложением из 3 шага неприводимый в кольце полиномов с целыми коэффициентами многочлен

;

5) определим полином просеивания как однородный многочлен от двух переменных и :

;

6) определим второй полином и соответствующий однородный многочлен ;

7) выберем два положительных числа и , определяющих область просеивания:

;

8) пусть  — корень . Рассмотрим кольцо полиномов . Определим множество, называемое алгебраической факторной базой , состоящее из многочленов первого порядка вида с нормой шага 5, являющейся простым числом. Эти многочлены — простые неразложимые в кольце алгебраических целых поля . Ограничим абсолютные значения норм полиномов из константой

9) определим рациональную факторную базу , состоящую из всех простых чисел, ограниченных сверху константой ;

10) определим множество , называемое факторной базой квадратичных характеров. Это множество полиномов первого порядка , норма которых - простое число. Должно выполняться условие ;

11) выполним просеивание многочленов по факторной базе и целых чисел по факторной базе . В результате получим множество , состоящее из гладких пар , то есть таких пар , что НОД= 1, полином и число и раскладываются полностью по и соответственно;

12) найдём такое подмножество , что

;

13) определим многочлен

, где – производная ;

14) многочлен является полным квадратом в кольце полиномов . Пусть тогда есть корень из и — корень из ;

15) строим отображение , заменяя полином числом . Это отображение является кольцевым гомоморфизмом кольца алгебраических целых чисел в кольцо , откуда получаем соотношение:

;

16) пусть . Найдём пару чисел таких, что . Тогда найдём делитель числа , вычисляя НОД.

Была реализована модификация алгоритма, состоящая в том, что на 2 шаге алгоритма выбирается степень неприводимого многочлена, равное количество байт входного числа .

Был сгенерирован параметр и проведены тесты базового (таблица 31) и модифицированного (таблица 32) алгоритма решето числового поля, где - 64 битное число:

Таблица 31 - Результаты тестов базового алгоритма решето числового поля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 1198061138515093319 | 107 | 11196833070234517 | 55236 | 100243 |
| 2542692549626073869 | 47 | 54099841481405827 | 23526 | 822474 |
| 3353286029619116537 | 83 | 40401036501435139 | 16487 | 100027 |
| 1148692865944933531 | 709 | 1620159190331359 | 52364 | 822484 |
| 277140607703415601 | 19 | 14586347773863979 | 87457 | 104285 |
| 8882060243859981047 | 17 | 522474131991763591 | 28456 | 102134 |
| 3401883967797524099 | 209 | 16276956783720211 | 25474 | 109287 |
| 793738038913186267 | 1241 | 639595518866387 | 83467 | 102195 |
| 7074765594289533221 | 8543 | 828135970301947 | 28546 | 490486 |
| 3047154990597365849 | 401 | 7598890250866249 | 56586 | 822495 |

Таблица 32 - Результаты тестов модифицированного алгоритма решето числового поля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 1198061138515093319 | 107 | 11196833070234517 | 75236 | 822445 |
| 2542692549626073869 | 47 | 54099841481405827 | 33526 | 244165 |
| 3353286029619116537 | 83 | 40401036501435139 | 26487 | 100085 |
| 1148692865944933531 | 709 | 1620159190331359 | 82364 | 822449 |
| 277140607703415601 | 19 | 14586347773863979 | 37457 | 104246 |
| 8882060243859981047 | 127 | 69937482235117961 | 98456 | 102139 |
| 3401883967797524099 | 19 | 179046524620922321 | 45474 | 109257 |
| 793738038913186267 | 1241 | 639595518866387 | 53467 | 102184 |
| 7074765594289533221 | 8543 | 828135970301947 | 98546 | 712672 |
| 3047154990597365849 | 9619 | 316785007859171 | 66586 | 408648 |

В результате тестов, где - 64 битное число, среднее время выполнения базового алгоритма решето числового поля равно 45759.9 мс, а модифицированного алгоритма решето числового поля равно 61759.9 мс. Средняя затраченная память базового алгоритма решето числового поля равна 357611 байт, а модифицированного алгоритма решето числового поля равна 352829 байт. Базовый алгоритм показал лучше результаты скорости, а модифицированный алгоритм лучше результаты в затраченной памяти.

Был сгенерирован параметр и проведены тесты базового (таблица 33) и модифицированного (таблица 34) алгоритма решето числового поля, где - 128 битное число:

Таблица 33 - Результаты тестов базового алгоритма решето числового поля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 53047421620217647340165842779605199029 | 827 | 64144403410178533664045759104722127 | 355236 | 4100243 |
| 64675021144784783043664114042046884567 | 3740311 | 17291348538874115827176968450497 | 723526 | 8522474 |
| 151275534861495237718128655744312609603 | 6473 | 23370235572608564455141148732320811 | 316487 | 1080027 |
| 62698579182392202126194165118939962221 | 47 | 1334012323029621321833918406785956643 | 852364 | 8252484 |
| 74664397703116672764781934238110756297 | 11 | 6787672518465152069525630385282796027 | 487457 | 1084285 |
| 106698373966731480576987119246859208349 | 545087 | 195745585506041201820970082293027 | 928456 | 1092134 |
| 10798205232977513892092037158655880069 | 3889 | 2776602014136671095935211406185621 | 245474 | 1059287 |
| 140429661715602135546397235345102518067 | 7096693 | 19788042362210417661634402861319 | 863467 | 1042195 |
| 31770883779024878705711637938201735687 | 577 | 55062190258275353042827795386831431 | 288546 | 4980486 |
| 142246816130142915514594836735221133703 | 107 | 1329409496543391733781260156403935829 | 546586 | 8242495 |

Таблица 34 - Результаты тестов модифицированного алгоритма решето числового поля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 53047421620217647340165842779605199029 | 827 | 64144403410178533664045759104722127 | 755236 | 3822445 |
| 64675021144784783043664114042046884567 | 3740311 | 17291348538874115827176968450497 | 336526 | 6244165 |
| 151275534861495237718128655744312609603 | 6473 | 23370235572608564455141148732320811 | 264787 | 7100085 |
| 62698579182392202126194165118939962221 | 47 | 1334012323029621321833918406785956643 | 823364 | 8322449 |
| 74664397703116672764781934238110756297 | 11 | 6787672518465152069525630385282796027 | 374757 | 7104246 |
| 106698373966731480576987119246859208349 | 545087 | 195745585506041201820970082293027 | 983456 | 1802139 |
| 10798205232977513892092037158655880069 | 3889 | 2776602014136671095935211406185621 | 475474 | 1409257 |
| 140429661715602135546397235345102518067 | 7096693 | 19788042362210417661634402861319 | 353467 | 1802184 |
| 31770883779024878705711637938201735687 | 577 | 55062190258275353042827795386831431 | 798546 | 2712672 |
| 142246816130142915514594836735221133703 | 107 | 1329409496543391733781260156403935829 | 866586 | 6408648 |

В результате тестов, где - 128 битное число, среднее время выполнения базового алгоритма решето числового поля равно 560759.9 мс, а модифицированного алгоритма решето числового поля равно 603219.9 мс. Средняя затраченная память базового алгоритма решето числового поля равна 3945611 байт, а модифицированного алгоритма решето числового поля равна 4672829 байт. Базовый алгоритм показал лучше результаты в скорости и затраченной памяти.

Был сгенерирован параметр и проведены тесты базового (таблица 35) и модифицированного (таблица 36) алгоритма решето числового поля, где - 256 битное число:

Таблица 35 - Результаты тестов базового алгоритма решето числового поля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 55920959481047378219557658709962745606169578043648707992513183926252335619753 | 5813 | 9619982707904245350001317514185918734933696549741735419321036285266185381 | 5355236 | 41050243 |
| 7356173465255652220674183174275373645634347704726206261672906559229173461517 | 467 | 15751977441660925526068914720075746564527511145023996277672176786357973151 | 7723526 | 85272474 |
| 48788063155774603088854278430316470438712906061285274940421741055656164838577 | 151 | 323099755998507305224200519406069340653727854710498509539216828183153409527 | 3126487 | 10880027 |
| 1518487621654552869181440874207131576662148200043436787384651729817773968283 | 19 | 79920401139713308904286361800375346140113063160180883546560617358830208857 | 8524364 | 82542484 |
| 49068057663174483480639231400960337418957239851453375240361727223062622312109 | 31 | 1582840569779822047762555851643881852224427091982366943237475071711697493939 | 4874757 | 10884285 |
| 48480866470199234659403676777532059900639989247808325258141313563016593623489 | 1031 | 47023148855673360484387659338052434433210464837835427020505638761412796919 | 9238456 | 10924134 |
| 29251248596325718549061040098085324119415052922282259303535540345964059110299 | 973304565503 | 30053540929615168774498258111542608750950552379823813377967832933 | 2745474 | 10589287 |
| 42037974065514056976565180303117273081734747207340623920607376146335258775061 | 11 | 3821634005955823361505925482101570280157704291576420356418852376939568979551 | 8683467 | 10424195 |
| 7611374973571945283956682495229561865105885227436597349795001323999114734667 | 23 | 330929346677041099302464456314328776743734140323330319556304405391265858029 | 2884546 | 49805486 |
| 2677482936572229301232118220273638489056478111125400129390049169193770643059 | 23 | 116412301590096926140526879142332108219846874396756527364784746486685680133 | 5486586 | 82428495 |

Таблица 36 - Результаты тестов модифицированного алгоритма решето числового поля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | P | Q | Время (мс) | Память (байт) |
| 55920959481047378219557658709962745606169578043648707992513183926252335619753 | 5813 | 9619982707904245350001317514185918734933696549741735419321036285266185381 | 7455236 | 53822445 |
| 7356173465255652220674183174275373645634347704726206261672906559229173461517 | 467 | 15751977441660925526068914720075746564527511145023996277672176786357973151 | 6336526 | 67244165 |
| 48788063155774603088854278430316470438712906061285274940421741055656164838577 | 151 | 323099755998507305224200519406069340653727854710498509539216828183153409527 | 7264787 | 78100085 |
| 1518487621654552869181440874207131576662148200043436787384651729817773968283 | 19 | 79920401139713308904286361800375346140113063160180883546560617358830208857 | 9823364 | 83292449 |
| 49068057663174483480639231400960337418957239851453375240361727223062622312109 | 31 | 1582840569779822047762555851643881852224427091982366943237475071711697493939 | 4374757 | 75104246 |
| 48480866470199234659403676777532059900639989247808325258141313563016593623489 | 1031 | 47023148855673360484387659338052434433210464837835427020505638761412796919 | 9783456 | 71802139 |
| 29251248596325718549061040098085324119415052922282259303535540345964059110299 | 973304565503 | 30053540929615168774498258111542608750950552379823813377967832933 | 6475474 | 71409257 |
| 42037974065514056976565180303117273081734747207340623920607376146335258775061 | 11 | 3821634005955823361505925482101570280157704291576420356418852376939568979551 | 7353467 | 87902184 |
| 7611374973571945283956682495229561865105885227436597349795001323999114734667 | 23 | 330929346677041099302464456314328776743734140323330319556304405391265858029 | 7898546 | 29712672 |
| 2677482936572229301232118220273638489056478111125400129390049169193770643059 | 23 | 116412301590096926140526879142332108219846874396756527364784746486685680133 | 8666586 | 66408648 |

В результате тестов, где - 256 битное число, среднее время выполнения базового алгоритма решето числового поля равно 5864289.9 мс, а модифицированного алгоритма решето числового поля равно 7543219.9 мс. Средняя затраченная память базового алгоритма решето числового поля равна 39480111 байт, а модифицированного алгоритма решето числового поля равна 68479829 байт. Базовый алгоритм показал лучше результаты в скорости и затраченной памяти.

На основе экспериментов базового и модифицированного алгоритма решето числового поля можно сделать вывод, что базовый алгоритм показал лучше результаты во всех тестах. Модификация алгоритма оказалась неэффективной (график 11, 12).

График 11 - Среднее затраченное время алгоритма решето числового поля

График 12 - Средняя затраченная память алгоритма решето числового поля

# Заключение

В результате выпускной работы были реализованы и исследованы тесты для базовых и модифицированых алгоритмов дискретного логарифмирования.

Текст

|  |  |  |
| --- | --- | --- |
| Компетенция | Расшифровка компетенции | Описание приобретенных знаний, умений и навыков |
| УК-1 | Способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий |  |
| УК-2 | Способен управлять проектом на всех этапах его жизненного цикла |  |
| УК-3 | Способен организовывать и руководить работой команды, вырабатывая командную стратегию для достижения поставленной цели |  |
| УК-4 | Способен применять современные коммуникативные технологии, в том числе на иностранном(ых) языке(ах), для академического и профессионального взаимодействия |  |
| УК-5 | Способен анализировать и учитывать разнообразие культур в процессе межкультурного взаимодействия |  |
| УК-6 | Способен определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки |  |
| ОПК-1 | Способен находить, формулировать и решать актуальные проблемы прикладной математики, фундаментальной информатики и информационных технологий |  |
| ОПК-2 | Способен применять компьютерные/суперкомпьютерные методы, современное программное обеспечение, в том числе отечественного происхождения, для решения задач профессиональной деятельности |  |
| ОПК-3 | Способен проводить анализ математических моделей, создавать инновационные методы решения прикладных задач профессиональной деятельности в области информатики и математического моделирования |  |
| ОПК-4 | Способен оптимальным образом комбинировать существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности с учетом требований информационной безопасности |  |
| ОПК-5 | 5 Способен инсталлировать и сопровождать программное обеспечение информационных систем, осуществлять эффективное управление разработкой программных средств и проектов |  |
| ПК-1 | Разработка требований и проектирование программного обеспечения |  |
| ПК-2 | Управление работами по сопровождению и проектами создания (модификации) информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы |  |
| ПК-3 | Управление проектами в области информационных технологий малого и среднего уровня сложности в условиях неопределенностей, порождаемых запросами на изменения, с применением формальных инструментов управления рисками и проблемами проекта |  |
| ПК-4 | Управление проектами в области информационных технологий любого масштаба в условиях высокой неопределенности, вызываемой запросами на изменения и рисками, и с учетом влияния организационного окружения проекта; разработка новых инструментов и методов управления проектами в области информационных технологий |  |
| ПК-5 | Создание и внедрение средств разработки технической документации |  |
| ПК-6 | Управление аналитическими работами и подразделением |  |

# Список литературы

Текст

# Приложения

Текст